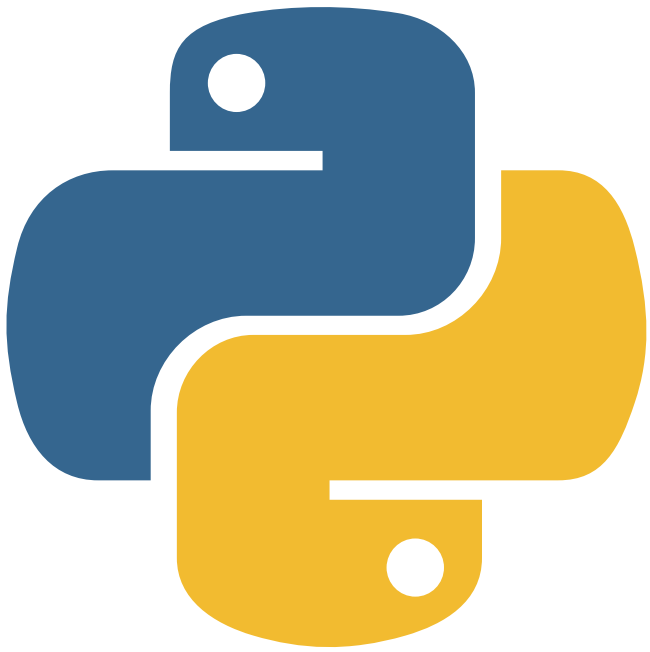
Day 4 : Python Basics to Advanced

Chapter 4 : Operators in Python

Todays Topic:

- **Arithmetic operators**
- **Comparison operators**
- **Assignment Operators**
- **Logical Operators**
- **Bitwise Operators**
- **Membership Operators**
- **Identity Operators**
- **Arithmetic Operators**

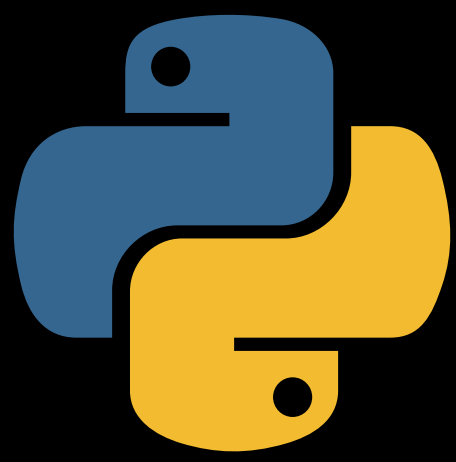


By @Curious_.programmer

4. Operators in Python

Made By:

Yadneyesh (Curious Coder)
CodWithCurious.com



4.1 Operators in Python

Operators in Python are symbols or special characters that are used to perform specific operations on variables and values.

Python provides various types of operators to manipulate and work with different data types. Here are some important categories of operators in Python:

- Arithmetic operators
- Comparison operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators
- Arithmetic Operators

Arithmetic Operators:

Arithmetic operators in Python are used to perform mathematical calculations on numeric values. The basic arithmetic operators include:

- **Addition (+)**: Adds two operands together. For example, if we have a = 10 and b = 10, then a + b equals 20.

- **Subtraction (-)**: Subtracts the second operand from the first operand. If the first operand is smaller than the second operand, the result will be negative. For example, if we have a = 20 and b = 5, then a - b equals 15.

- **Division (/)**: Divides the first operand by the second operand and returns the quotient. For example, if we have a = 20 and b = 10, then a / b equals 2.0.

- **Multiplication (*)**: Multiplies one operand by the other. For example, if we have a = 20 and b = 4, then a * b equals 80.

- **Modulus (%)**: Returns the remainder after dividing the first operand by the second operand. For example, if we have a = 20 and b = 10, then a % b equals 0.

- **Exponentiation (**) or Power:** Raises the first operand to the power of the second operand. For example, if we have a = 2 and b = 3, then a ** b equals 8.

- **Floor Division (//):** Provides the floor value of the quotient obtained by dividing the two operands. It returns the largest integer that is less than or equal to the result. For example, if we have a = 20 and b = 3, then a // b equals 6.

Comparison Operators:

Comparison operators in Python are used to compare two values and return a Boolean value (True or False) based on the comparison. Common comparison operators include:

- Equal to (==): Checks if two operands are equal.

- Not equal to (!=): Checks if two operands are not equal.

- Greater than (>): Checks if the left operand is greater than the right operand.

- Less than (<): Checks if the left operand is less than the right operand.

- Greater than or equal to (>=): Checks if the left operand is greater than or equal to the right operand.

- Less than (<): Checks if the left operand is less than the right operand.

- Greater than or equal to (>=): Checks if the left operand is greater than or equal to the right operand.

- Less than or equal to (<=): Checks if the left operand is less than or equal to the right operand.

Assignment Operators:

Assignment operators are used to assign values to variables. They include:

- Equal to (=): Assigns the value on the right to the variable on the left.

- Compound assignment operators (+=, -=, *=, /=): Perform the specified arithmetic operation and assign the result to the variable.

Logical Operators:

Logical operators in Python are used to perform logical operations on Boolean values. The main logical operators are:

- Logical AND (and): Returns True if both operands are True, otherwise False.

- Logical OR (or): Returns True if at least one of the operands is True, otherwise False.

- Logical NOT (not): Returns the opposite Boolean value of the operand.

Bitwise Operators:

Bitwise operators perform operations on individual bits of binary numbers. Some common bitwise operators in Python are:

- Bitwise AND (&): Performs a bitwise AND operation on the binary representations of the operands.

- Bitwise OR (|): Performs a bitwise OR operation on the binary representations of the operands.

- Bitwise XOR (^): Performs a bitwise exclusive OR operation on the binary representations of the operands.

- Bitwise complement (~): Inverts the bits of the operand.
- Left shift (<<): Shifts the bits of the left operand to the left by the number of positions specified by the right operand.

- Right shift (>>): Shifts the bits of the left operand to the right by the number of positions specified by the right operand.

- Right shift (>>): Shifts the bits of the left operand to the right by the number of positions specified by the right operand.

Membership Operators:

Membership operators are used to test whether a value is a member of a sequence (e.g., string, list, tuple). They include:

- In: Returns True if the value is found in the sequence.

- Not in: Returns True if the value is not found in the sequence.

Identity Operators:

Identity operators are used to compare the identity of two objects. They include:

- Is: Returns True if both operands refer to the same object.

- Is not: Returns True if both operands do not refer to the same object.

All Coding Handwritten Notes

for Handwritten Notes Search
“CodeWithCurious Store” on Google

Search

CodeWithCurious Store

@curious_.programmer
CodeWithCurious.com

Day 5 Control Flow

Will Post Follow For More

Want This PDF?

Join Our Telegram and Download 100+ PDFs and Handwritten Notes